

OPTIMAL ALGORITHMS OF GRAM-SCHMIDT TYPE

JAMES B. WILSON

ABSTRACT. Three algorithms of Gram-Schmidt type are given that produce an orthogonal decomposition of finite d -dimensional symmetric, alternating, or Hermitian forms over division rings. The first uses $d^3/3 + O(d^2)$ ring operations with very simple implementation. Next, that algorithm is adapted in two new directions. One is an optimal sequential algorithm whose complexity matches the complexity of matrix multiplication. The other is a parallel NC algorithm with similar complexity.

1. INTRODUCTION

The classic Gram-Schmidt ‘orthogonalization process’ returns an orthonormal basis of an inner product space. Here we generalize that process in the appropriate fashion to Hermitian forms over division rings Δ . For us a Hermitian Δ -form is a function $b : V \times V \rightarrow \Delta$ on a finite-dimensional Δ -vector space V where b is linear in the first variable and for some anti-isomorphism σ of Δ , for all $u, v \in V$, $b(u, v) = b(v, u)^\sigma$. This captures the usual symmetric and skew-symmetric forms as well as the traditional Hermitian forms; cf [10]. We identify V with a space of row vectors and so describe b by a matrix B such that $b(u, v) = uBv^{\sigma t}$. The assumptions on b force $B = 0$, or $B = sB^{\sigma t}$ with $s = \pm 1$ and $\sigma^2 = 1$. To change the basis we use an invertible matrix A and observe $b(uA, vA) = uABA^{\sigma t}v^{\sigma t}$. Hence, a fully refined orthogonal decomposition for b is captured by a matrix A under which $ABA^{\sigma t}$ is nearly diagonal, nearly in that sometimes $J := \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ is required.

Theorem 1. *Let Δ be a division ring, let $s = \pm 1$, and let σ be a unital anti-isomorphism of Δ with $\sigma^2 = 1$. There are deterministic algorithms that, given a $(d \times d)$ -matrix $B = sB^{\sigma t}$, return an invertible $(d \times d)$ -matrix A such that*

$$(1.1) \quad ABA^{\sigma t} = B_1 \oplus \cdots \oplus B_m$$

and each B_i is either 1×1 or J .

- (i) *The first algorithm uses e inversions, $\binom{d}{2}$ equality tests, $e^3/3 + O(d^2)$ additions and $e^3/3 + O(d^2)$ multiplications in Δ , where $e = d - r$ and r is the rank of the null space of B .*
- (ii) *The second algorithm returns a straight line program to A using $O(d^\omega)$ operations in Δ , where ω is the exponent of matrix multiplication.*
- (iii) *The third algorithm is parallel NC³ in an arithmetic model, that is, it uses $O(\log^3 d)$ operations in Δ on $d^{O(1)}$ processors.*

Date: January 21, 2011.

Key words and phrases. bilinear form, sesquilinear form, Hermitian form, polynomial-time algorithm.

Theorem 1 in part proves that Hermitian forms over division rings have a decomposition of the type described in (1.1). For fields this is well-known, e.g. [1, Theorems 3.7], but most proofs begin by classifying forms into subclasses, e.g. symmetric if $s = 1$ and $\sigma = 1$, skew-symmetric if $s = -1$ and $\sigma = 1$, or Hermitian if $\sigma \neq 1$. Nice bases are constructed by individual arguments for each case. Here we find a single argument allows for uniform optimal asymptotic and parallel algorithms without dependence on Δ .

The idea behind Theorem 1(i) is shared by many generalizations of Gram-Schmidt. For symmetric forms it goes back at least to Smiley's *Algebra of Matrices* [8, Section 12.2] and is adequately described as symmetric Gaussian elimination. Dax and Kaniel [3] give a detailed analysis of such an algorithm for symmetric forms. Holt and Roney-Dougal [4] use the method in a case-by-case algorithm for Hermitian forms over finite fields. I was also gratefully alerted to a predecessor to Theorem 1(ii) that applies to symmetric forms over fields; see [2, Theorem 16.25].

The algorithms for Theorem 1 parts (ii) and (iii) settle the complexity of finding an nice basis for a generic Hermitian form, but these may not be best suited for certain applications. First, they depend on data structures for fast matrix multiplication which may provide an undesirable overhead in small dimensions. The exact cross-over dimension is an issue of ongoing research; see [11, p. 313]. Furthermore, our algorithms assume exact field operations, such as in algebraic number fields, rational Quaternion division rings, or finite fields. We make no claims about their numerical stability in fields with floating point approximations. In such cases consider [6].

Each of our algorithms allows the user to choose a computational encoding for Δ , such as by polynomials or matrices over a field. If no alternative suggests itself, an adequate method is to encode Δ by structure constants over its center; see [7, p. 223]. Also, b can be encoded as a “black-box”; however, we will eventually evaluate b on all unordered pairs from a fixed basis for V and so it simplifies our description to assume that b is input by a $(d \times d)$ -matrix $B = sB^{\sigma t}$ with $s = \pm 1$ and $\sigma^2 = 1$. If the s and σ are not specified with B , then suitable values can be detected during the execution of the algorithms for Theorem 1. We either prove that $B = 0$ or we find $u, v \in V$ such that $b(u, v) = uBv^{\sigma t} \neq 0$. The first such pair $u, v \in V$ determines σ by $\sigma : \alpha \mapsto b(u, \alpha v)b(u, v)^{-1}$, and $s = b(v, u)^{-1} \cdot b(u, v)^{\sigma}$. We write the algorithm as though s and σ are known.

2. SMILEY'S METHOD

Let us start with the algorithm **Decompose** which is not asymptotically optimal, but which (I believe) is the simplest to implement and captures all Hermitian forms at once. This is the prototype for the optimal sequential and parallel algorithms given later.

If A is an elementary matrix then $ABA^{\sigma t}$ modifies B in one of three ways. First, if A is a diagonal matrix with 1's on the diagonal except for λ in entry i , then $ABA^{\sigma t}$ scales row i by λ , and column i by λ^{σ} . For instance:

$$\begin{bmatrix} 1 & & \\ & \lambda & \\ & & 1 \end{bmatrix} \begin{bmatrix} \alpha & \beta & \delta \\ s\beta^{\sigma} & \gamma & \epsilon \\ s\delta^{\sigma} & s\epsilon^{\sigma} & \phi \end{bmatrix} \begin{bmatrix} 1 & & \\ & \lambda & \\ & & 1 \end{bmatrix}^{\sigma t} = \begin{bmatrix} \alpha & \beta\lambda^{\sigma} & \delta \\ s(\beta\lambda^{\sigma})^{\sigma} & \lambda\gamma\lambda^{\sigma} & \lambda\epsilon \\ s\delta^{\sigma} & s(\lambda\epsilon)^{\sigma} & \phi \end{bmatrix}$$

We describe that as *scaling row-column i by λ* . Second, if A is a transposition of i and j then $ABA^{\sigma t}$ has the entries from B with rows i and j swapped as well as columns i and j swapped. We call this *swapping row-column i with row-column j* . That does not involve operations in Δ . Thirdly, if $A = I + \lambda E_{ij}$ then $ABA^{\sigma t}$ has the effect of adding λ times row i to row j and λ^σ times column i to column j , as illustrated below.

$$\begin{bmatrix} 1 & & \\ & 1 & \\ -\gamma^\sigma & & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & \gamma \\ s & \beta & \delta \\ s\gamma^\sigma & s\delta^\sigma & \epsilon \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ -\gamma^\sigma & & 1 \end{bmatrix}^{\sigma t} = \begin{bmatrix} 0 & 1 & 0 \\ s & \beta & \delta - \beta\gamma \\ 0 & s(\delta - \beta\gamma)^\sigma & * \end{bmatrix}$$

That implicitly involved the fact that entries β on the diagonal satisfy $\beta = s\beta^\sigma$. To *clear a row-column* means to use a selected non-zero entry j in a row-column i of B , and use successive multiplications by $I + \lambda_k E_{ki}$, for $k \in \{1, \dots, d\} - \{i\}$ to set all other entries in the row-column i to zero. This is possible whenever $i = j$ or $B_{ii} = 0$. Using the symmetry of the matrices $B = sB^{\sigma t}$, clearing a row-column uses $d^2 + O(d)$ additions, $d^2 + O(d)$ multiplications, d applications of σ , and one inversion.

We use upper case Roman letters for block sub-matrices and lower case Greek letters for coefficients in Δ . We also assume that the associated matrix A which transforms B into the return $ABA^{\sigma t}$ as in (1.1) is evident from the operations described, and so we do not explicitly include A in the description of the algorithm.

Standardize $\left(B = \begin{bmatrix} 0 & 1 \\ s & \alpha \end{bmatrix} \right)$:

If $\alpha \neq 0$, set $A = \begin{bmatrix} 1 & -\alpha^{-1} \\ 0 & 1 \end{bmatrix}$ and return $ABA^{\sigma t} = [-s\alpha^{-1}] \oplus [\alpha]$. If $\alpha = 0$

and $s = 1 \neq -1$, set $A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and return $ABA^{\sigma t} = [2] \oplus [-2]$. Else return B .

Decompose $(B \in M_d(\Delta) : B = sB^{\sigma t})$:

(I) **[Base case]** If $d \leq 1$ return B .

(II) **[Anisotropic case]** If $B_{11} = \beta \neq 0$, then use that entry to clear the remaining non-zero entries of row-column 1. Now $B = \begin{bmatrix} \beta & 0 \\ 0 & B' \end{bmatrix}$.

Return $[\beta] \oplus \text{Decompose}(B')$.

(III) **[Isotropic case]** Else, if $B_{12} = \gamma \neq 0$ (after a possible swap of a

row-column), i.e. $B = \begin{bmatrix} 0 & \gamma & * \\ s\gamma^\sigma & \alpha & * \\ * & * & * \end{bmatrix}$, then scale row-column 2 by γ^{-1} and

excluding B_{22} , use B_{12} to clear row-column 1 and B_{21} to clear row-column

2. Now $B = \begin{bmatrix} B' & 0 \\ 0 & B'' \end{bmatrix}$ where $B' = \begin{bmatrix} 0 & 1 \\ s & \alpha \end{bmatrix}$.

Return **Standardize** $(B') \oplus \text{Decompose}(B'')$.

(IV) **[Radical case]** Else, $B = \begin{bmatrix} 0 & 0 \\ 0 & B' \end{bmatrix}$ so return $[0] \oplus \text{Decompose}(B')$.

Proof of Theorem 1(i). The algorithm **Decompose** returns a block diagonal matrix whose blocks are as in (1.1). That algorithm only modifies the entries of B so that the space complexity is $O(d^2)$ elements in Δ .

Now we consider the time complexity. There are at most d equality tests to decide on the correct case to enter. The anisotropic case clears one row-column and recurses on a matrix of dimension $d - 1$. The isotropic case clears two row-columns, performs some multiplications of (2×2) -matrices, and recurses on a matrix of dimension $d - 2$. Finally, the radical case simply recurses on a matrix of dimension $d - 1$. Hence, if $T(d)$ is the number of additions performed by the algorithm, then $T(d) \in 2d^2 + T(d - 2) + O(d)$. If r is the dimension of the radical and $e = d - r$, then $T(d) \in e^3/3 + O(d^2)$. The algorithm uses the same number of multiplications, $\binom{d}{2}$ equality tests, e inversions, and $\binom{d}{2} - \binom{r}{2}$ applications of σ . \square

3. OPTIMAL AND PARALLEL METHODS

Multiplication of $(d \times d)$ -matrices by the traditional algorithm is not the most efficient method for large dimensions. The various new methods use $O(d^\omega)$ operations in Δ for some $2 \leq \omega \leq 3$ [11, p. 315]. Here we prove the same complexity for finding a decomposition as in (1.1). Bürgisser et. al. give an example of a symmetric $(d \times d)$ -matrix over a field where the complexity of finding an orthogonal basis is $O(d^\omega)$ (provided that $\omega > 2$) [2, Theorem 16.20] and so the complexity in Theorem 1(ii) is best possible in general.

Proof of Theorem 1(ii). The algorithm **DecomposeByBlocks** suffices as described so it remains to analyze the time complexity of the algorithm.

We start by detecting the radical of B . This amounts to solving for a basis of the null space of B . That has complexity of $O(d^\omega)$ [9, Theorem 2]. To create B'' requires 2 matrix multiplications and d^2 applications of σ . Thus the radical case uses $O(d^\omega)$ operations in Δ . The algorithm never re-enters this case.

In the block anisotropic case we solve for a null space on a $\lceil d/2 \rceil$ -square matrix B' , and multiply two $(d \times d)$ -matrices, in (3.1). Let f be the rank of the null space of B' . At this point we have two cases. If $f = \lceil d/2 \rceil$ we exit the block anisotropic case and enter the block semi-hyperbolic case; otherwise, we to create Y (we invert and multiply a $((\lceil d/2 \rceil - f) \times (\lceil d/2 \rceil - f))$ -matrix), multiply two $(d \times d)$ -matrices in (3.2). We then make one recurse call to the block anisotropic case for B'' , and one call to the block semi-hyperbolic case for $\begin{bmatrix} 0 & X \\ sX^{\sigma t} & Z \end{bmatrix}$ where X has f rows and Z is $(d - \lceil d/2 \rceil) \times (d - \lceil d/2 \rceil)$. Ignoring the recursions, the anisotropic case uses $O(d^\omega)$ operations in Δ .

The block semi-hyperbolic case takes in a $(d \times d)$ -matrix partitioned by into $(f, d - f)$ -blocks. We compute a null column space of an $(f \times (d - f))$ -matrix, multiply 2 $(d \times d)$ -matrices (3.3) ((3.4) requires no computation), and we also multiply two $(2f \times 2f)$ -matrices in (3.5). Finally there are at most $d/2$ applications of **Standardize** and a recursive call on a $((d - 2f) \times (d - 2f))$ -matrix B' . All this amounts to $O(d^\omega)$ operations in Δ before the recursion.

Now we estimate the total cost. Let $T_a(d)$ be the cost of the block anisotropic case for an input of dimension d , and $T_h(d, f)$ the cost of the block semi-hyperbolic case for an input of dimension d where X has f -rows. For some constants $C_a, C_h > 0$,

DecomposeByBlocks($B \in M_d(\Delta) : B = sB^{\sigma t}$):

(I) [**Detect Radical**] Compute an invertible A such that $AB = \begin{bmatrix} B' \\ 0 \end{bmatrix}$ where

B' has full row rank. Now $ABA^{\sigma t} = \begin{bmatrix} B'' & 0 \\ 0 & 0 \end{bmatrix}$ with B'' nonsingular. Apply step (II) to B'' .

(II) [**Block Anisotropic case**] Here B is a nonsingular $(d \times d)$ -matrix. If $d \leq 1$, halt; else take $B = \begin{bmatrix} B' & * \\ * & * \end{bmatrix}$ with $B' \in M_{\lceil d/2 \rceil}(\Delta)$. Find A such that

$AB'A^{\sigma t} = \begin{bmatrix} B'' & 0 \\ 0 & 0 \end{bmatrix}$ and B'' has full rank (as in (I)). Compute

$$(3.1) \quad \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} B' & * \\ * & * \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}^{\sigma t} = \begin{bmatrix} AB'A^{\sigma t} & * \\ * & * \end{bmatrix} = \begin{bmatrix} B'' & 0 & C \\ 0 & 0 & W \\ sC^{\sigma t} & sW^{\sigma t} & * \end{bmatrix}.$$

If B'' has dimension 0 then (as B is nonsingular) W is nonsingular; proceed to step (III). Otherwise, $B'' = s(B'')^{\sigma t}$ is nonsingular. Set $Y = -sC^{\sigma t}(B'')^{-1}$ and compute

$$(3.2) \quad \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ Y & 0 & I \end{bmatrix} \begin{bmatrix} B'' & 0 & C \\ 0 & 0 & * \\ sC^{\sigma t} & * & * \end{bmatrix} \begin{bmatrix} I & 0 & Y^{\sigma t} \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} B'' & 0 & 0 \\ 0 & 0 & X \\ 0 & sX^{\sigma t} & Z \end{bmatrix}.$$

Note X has full row rank since B is nonsingular. Apply step (II) to B'' , and apply step (III) to $\begin{bmatrix} 0 & X \\ sX^{\sigma t} & Z \end{bmatrix}$; then halt.

(III) [**Block Isotropic case**] Now $B = \begin{bmatrix} 0 & X \\ sX^{\sigma t} & * \end{bmatrix}$ and X has full row rank.

Compute an invertible matrix A such that $XA = \begin{bmatrix} C & 0 \end{bmatrix}$ where C has full column rank; thus, C is invertible. Compute

$$(3.3) \quad \begin{bmatrix} C^{-1} & 0 \\ 0 & A^{\sigma t} \end{bmatrix} \begin{bmatrix} 0 & X \\ sX^{\sigma t} & * \end{bmatrix} \begin{bmatrix} C^{-\sigma t} & 0 \\ 0 & A \end{bmatrix} = \begin{bmatrix} 0 & I & 0 \\ sI & Z & Y \\ 0 & sY^{\sigma t} & B' \end{bmatrix}.$$

Observe that:

$$(3.4) \quad \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -sY^{\sigma t} & 0 & I \end{bmatrix} \begin{bmatrix} 0 & I & 0 \\ sI & Z & Y \\ 0 & sY^{\sigma t} & B' \end{bmatrix} \begin{bmatrix} I & 0 & -sY \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} 0 & I & 0 \\ sI & Z & 0 \\ 0 & 0 & B' \end{bmatrix}.$$

Let $B'' = \begin{bmatrix} 0 & I \\ sI & Z \end{bmatrix}$ and decompose $Z = sZ^{\sigma t} = U + D + sU^{\sigma t}$ where U is upper triangular with 0 entries on the diagonal. So D is diagonal and $D = sD^{\sigma}$. Reset B'' to be

$$(3.5) \quad \begin{bmatrix} I & 0 \\ -U & I \end{bmatrix} \begin{bmatrix} 0 & I \\ sI & Z \end{bmatrix} \begin{bmatrix} I & -U^{\sigma t} \\ 0 & I \end{bmatrix} = \begin{bmatrix} 0 & I \\ sI & D \end{bmatrix}.$$

Sort the row-columns so that the matrix is in the form

$\begin{bmatrix} 0 & 1 \\ s & \alpha_1 \end{bmatrix} \oplus \cdots \oplus \begin{bmatrix} 0 & 1 \\ s & \alpha_f \end{bmatrix}$ with $\alpha_i \in \Delta$. Apply **Standardize** to each of those blocks. Finally, B' is nonsingular so apply step (II) to B' , then halt.

$T_h(d, f) \leq T_a(d - 2f) + C_h d^\omega$, and

$$\begin{aligned} T_a(d) &\leq \max\{ T_h(d, d/2), T_a(d/2 - f) + T_h(d/2 + f, f) \} + C_a d^\omega \\ &\leq 2T_a(d/2 - f) + C_h(d/2 + f)^\omega + (C_a + C_h)d^\omega \\ &\leq 2T_a(d/2) + (C_a + 2C_h)d^\omega. \end{aligned}$$

Thus, $T(d) \in O(d^\omega)$. \square

Proof of Theorem 1(iii). **DecomposeByBlocks** uses $O(\log d)$ recursive calls and each step can use the parallel NC^2 (i.e. $O(\log^2 d)$) linear algebra algorithms of [5, Sections 3.8, 4.5] and [9, Section 2.3] to find null-spaces and multiply matrices in an arithmetic model. (Those methods make it possible to trade on time efficiency to reduce the number of required processors, which is of importance in practice.) \square

4. POST-PROCESSING ADJUSTMENTS

In our algorithms we opted for a decomposition of B which is as close to diagonal as possible so that the associated basis is nearly orthogonal. It is also common to want a decomposition with as many blocks of the form $J = \begin{bmatrix} 0 & 1 \\ s & 0 \end{bmatrix}$ as possible. The algorithm can be tuned in that direction by modifying **Standardize** and by converting various (1×1) -blocks into J 's at the end of the algorithm. The details are analogous to those used in **Standardize**.

In some cases a canonical return is possible with a few adjustments. For example, the block $[\alpha]$ can be adjusted to $[\gamma\alpha\gamma^\sigma]$ for $0 \neq \gamma \in \Delta$. Hence, if $\alpha = \gamma^{-1}\gamma^{-\sigma}$ for some $\gamma \in \Delta - \{0\}$, then we may replace α with 1. Computationally finding γ to perform this adjustment can be involved. Already when $\sigma = 1$ and Δ a field this amounts to finding a square-root of α . If the number of classes in Δ of the form $\gamma\alpha\gamma^\sigma$ is linearly ordered, it is possible to sort the (1×1) -blocks accordingly.

Another situation for modification tries to convert multiple (1×1) -blocks. For instance, if Δ is a field and $0 \neq \alpha = \gamma\gamma^\sigma + \delta\delta^\sigma$ for some $\gamma, \delta \in \Delta$ (for example, if $\sigma = 1$ and α is a sum of squares), then

$$\begin{bmatrix} \gamma & \delta \\ \delta^\sigma & -\gamma^\sigma \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma & \delta \\ \delta^\sigma & -\gamma^\sigma \end{bmatrix}^{\sigma t} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}.$$

Similarly, if the characteristic is 2 and $\alpha \neq 0$ then

$$\begin{bmatrix} 0 & \alpha & 1 \\ 1 & \alpha & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ & & \alpha \end{bmatrix} \begin{bmatrix} 0 & \alpha & 1 \\ 1 & \alpha & 1 \\ 1 & 0 & 1 \end{bmatrix}^{\sigma t} = \begin{bmatrix} \alpha & & \\ & \alpha & \\ & & \alpha \end{bmatrix}.$$

ACKNOWLEDGEMENTS

Thanks to Peter Brooksbank for suggesting this note and offering comments.

REFERENCES

- [1] E. Artin, *Geometric algebra*, Wiley Classics Library, John Wiley & Sons Inc., New York, 1988. Reprint of the 1957 original; A Wiley-Interscience Publication.
- [2] Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi, *Algebraic complexity theory*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 315, Springer-Verlag, Berlin, 1997. With the collaboration of Thomas Lickteig.
- [3] A. Dax and S. Kaniel, *Pivoting techniques for symmetric Gaussian elimination*, Numer. Math. **28** (1977), no. 2, 221–241.

- [4] Derek F. Holt and Colva M. Roney-Dougal, *Constructing maximal subgroups of classical groups*, LMS J. Comput. Math. **8** (2005), 46–79 (electronic).
- [5] Richard M. Karp and Vijaya Ramachandran, *Parallel algorithms for shared-memory machines*, Handbook of theoretical computer science, Vol. A, Elsevier, Amsterdam, 1990, pp. 869–941.
- [6] F. J. Linge, *Efficient Gram-Schmidt orthonormalisation on parallel computers*, Comm. Numer. Meth. Engng. **16** (2000), 57–66.
- [7] Lajos Rónyai, *Computations in associative algebras*, Groups and computation (New Brunswick, NJ, 1991), DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 11, Amer. Math. Soc., Providence, RI, 1993, pp. 221–243.
- [8] M. F. Smiley, *Algebra of matrices*, Allyn and Bacon, Inc., Boston, 1965.
- [9] V. I. Solodovnikov, *Upper bounds of complexity of the solution of systems of linear equations*, Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI) **118** (1982), 159–187, 215–216 (Russian, with English summary). The theory of the complexity of computations, I.
- [10] Donald E. Taylor, *The geometry of the classical groups*, Sigma Series in Pure Mathematics, vol. 9, Heldermann Verlag, Berlin, 1992.
- [11] Joachim von zur Gathen and Jürgen Gerhard, *Modern computer algebra*, 2nd ed., Cambridge University Press, Cambridge, 2003.

DEPARTMENT OF MATHEMATICS, THE OHIO STATE UNIVERSITY, COLUMBUS, OH 43210,
E-mail address: wilson@math.ohio-state.edu